CHOICE IS NOT JUST NON-DETERMINISM

[EXTENDED ABSTRACT]

RICHARD BUCKLAND*
Department of Computing, Macquarie University, Sydney
Distributed Research Group, Microsoft Institute, Sydney
AUSTRALIA
E-mail: richardb@macadam.mpce.mq.edu.au

1. Introduction

Of the established high level models of computation, including automata, CSP³, CCS⁶, Petri nets⁸, Mazurkiewicz traces⁸ and event structures⁹, none treats choices as individual entities. For each choice within a process the models express the fact that there is a choice and they record the alternative outcomes of the choice, but the actual choice — the decision procedure — is abstracted away.

This is entirely appropriate when describing processes at a high level of abstraction. However in the course of refining an abstract process into an actual program, or transforming a specification into incrementally more concrete specifications, the missing information needs to be incorporated. Furthermore in order to provide a rigorous treatment of the progression from the initial specification of a process to its implementation it is necessary that the same model of computation be used throughout.

The aims of this paper are twofold. Firstly to argue for an explicit representation of choice in process specifications, and secondly to develop an algebra of choice suitable for comparing and calculating with choices.

2. Making Choices Explicit

It is clear that it is necessary to incorporate choice information in process specifications at some stage — simply to be able to construct the process. However it is also important to provide some information about choices in a process specification even when the specification is a highly abstract one. This is because the behaviour of the process cannot be determined without knowing the interactions between its choices.

In particular if all choices are left unspecified then the process must be regarded as capable of a large number of traces: given by all possible combinations of the outcomes of all the choices. In actuality however the nature of process specification means that many of these combinations will not be possible⁵. The presence of such "impossible" traces makes it difficult to reason about liveness and safety properties such as deadlock.

3. Choices as objects

Choice has traditionally been treated as a second class citizen in comparison to transition. We can label transitions or leave them unlabelled, we can compare them, describe their interactions, and record when they are instances of the same underlying action. With choices, on the other hand, we typically only note their existence and in some cases group them into broad categories (e.g. internal vs external in CSP).

However in a process each choice is an individual entity, such as "if today is a weekday" or "if x = 0". At a more general level there are random non-deterministic choices about which nothing is known, probabilistic choices, choices determined by the environment, choices selected by the process itself and so on. Treating choices in the same manner as transitions would allow us to leave them unspecified when modelling at high levels of abstraction, and to denote the actual choice or the type of choice when modelling at lower levels of abstraction.

*The author gratefully acknowledges a Microsoft Fellowship and the support of the Australian Research Council and the British Council.
To treat choice in this manner we must have elements denoting choices in the process representation. Consider a traditional transition diagram. It is essentially a two-dimensional structure with states zero-dimensional elements, and transitions one-dimensional elements. The two-dimensional nature of the diagram arises from its branching. For each binary branch there is a particular choice that mediates between the one-dimensional alternatives. However there is no actual element of the graph denoting this choice — the presence of the choice is simply indicated by the branch, or rather, by the presence of a two-dimensional region between the branches.

In contrast higher dimensional transition systems (the constructive pasting schemes\(^2\), or sometimes simply schemes) can be used to denote choices. These are transition systems which include the regions between branches as explicit elements. Thus

\[
\begin{array}{c}
\begin{array}{c}
 a \quad b \\
 0 \quad 1 \\
 c 
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
 b \\
 0 \\
 e 
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
 c \\
 2 
\end{array}
\end{array}
\]

denotes the process which makes a choice \(\alpha\) between \(<a, b>\) and \(<c>\), whereas in a traditional transition system the corresponding diagram

\[
\begin{array}{c}
\begin{array}{c}
 a \quad b \\
 0 \quad 1 \\
 c 
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
 b \\
 0 \\
 e 
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
 c \\
 2 
\end{array}
\end{array}
\]

denotes the process which simply makes some choice between \(<a, b>\) and \(<c>\).

4. Denoting Processes

Here we briefly set out the link between a scheme and the process it denotes. Informally schemes can be viewed as directed graphs comprised of arrows (called cells) of various dimensions\(^3\). The 0-cells will denote states, 1-cells will denote the events which occur between 0-cells, and 2-cells the choices between sequences of 1-cells.

Compound choices are denoted in a straightforward manner:

\[
\begin{array}{c}
\begin{array}{c}
 b \quad f \\
 0 \quad 1 \\
 e 
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
 b \\
 0 \\
 e 
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
 f \\
 2 
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
 c \\
 3 
\end{array}
\end{array}
\]

is the process which initially makes choice \(x\) between the process which does \(<a>\), and the process which first does \(b\) then makes a choice \(y\) between \(<c>\) and \(<f, e>\). The 2-cells \(x, y\) thus jointly arbitrate between the three possible traces of the system.

A 3-cell will denote a meta-choice, that is a choice between two two-dimensional choices. Similarly 4-cells are meta-meta choices, and so on.

5. Specification = Flattening

It is evident that a high dimensional meta-choice can always be “flattened” to a collection of two-dimensional choices in the same way a collection of nested if's can be written as a single “flat” case statement. Similarly any higher dimensional n-scheme \(A_n\) can be flattened into some two-dimensional scheme \(A_2\).

It might seem then that the two representations \(A_n\) and \(A_2\) are equally useful and that the higher dimensional information of \(A_n\) is superfluous. However there is a sense in which \(A_n\) is a better representation for processes whose choices are not fully specified. This is because we have full information about the structure of \(A_n\) regardless of what its choices turn out to be; we have all the cells of \(A_n\), we know their dimension, their

\(^{2\text{Space does not permit a formal definition of constructive pasting schemes in this extended abstract. Complete definitions and examples of schemes are given in } 1^{4}.\)
orientation, and we know how they fit together. This is precisely the information about a process which we need to know to be able to combine it with other processes to form larger systems. On the other hand it can be shown that the structure of \( A_2 \) depends upon the as yet unspecified choices\(^2\), and hence we cannot combine \( A_2 \) with other processes until these have been fully specified.

Higher dimensional schemes are thus the fully abstract specifications of processes. Using this form nothing need be known about the particular events or choices of a process for it to be sequentially composed with other processes, for it to be combined with other processes in a choice, or for it to be concurrently executed alongside another process. As choice cells are specified and the processes become more concretely specified their schemes can be flattened to progressively lower dimensions.

The need for higher dimensional cells therefore arises from the desire to be able to work with processes of partially specified choices. Dimensionality provides the necessary algebraic support for choice refinement. This is the process of incrementally adding more information about particular choices to a specification. This is an intuitively pleasing result as it mirrors the known observation\(^7\) that dimensionality supports transition refinement.

6. An Algebra of Choice

We have argued for an explicit treatment of choice when specifying processes. To be of use such an ability to specify particular choices needs to be accompanied by the ability to calculate with choices — an algebra of choice.

This is necessary for two reasons. Firstly to detect the equivalence of specifications which are syntactically different, and secondly to calculate the flattened form of higher dimensional schemes as further detail is added to their specification.

We begin the presentation of the algebra by noting two properties of all choices\(^6\) Let \( A \circ_B B \) denote the process which makes a binary choice \( \times \) between \( A \) and \( B \). For any choice \( \circ_x \) we have idempotence:

\[
(A \circ_x A) = A
\]

and the existence of a unique dual choice \( \circ_x \) such that:

\[
(A \circ_x B) = (B \circ_x A).
\]

To illustrate duality: if \( \circ_x \) is select the first process if today is Saturday, the second process otherwise then \( \circ_x \) is select the second process if today is Saturday, the first process otherwise.

These are the only properties we claim for all choices. This may seem surprising given that in process algebras such as CCS and CSP choices are considered to be idempotent, associative, and commutative. It turns out that this is because process algebras only consider a subset of the possible choices which processes can exhibit. This claim is most simply supported by showing two counterexamples.

To see that not all choices are associative consider \( \circ_y = \text{select the first process with probability 50\%, the second otherwise.} \) Then

\[
((A \circ_y B) \circ_y C) \neq (A \circ_y (B \circ_y C))
\]

since in the former the probabilities of selecting \( A, B, \) and \( C \) are 25\%, 25\%, and 50\% respectively, whereas in the latter they are 50\%, 25\%, and 25\%.

To see that not all choices are symmetric consider \( \circ_x = \text{select the first process if } n \text{ is prime, the second otherwise.} \) It is evident that

\[
(A \circ_x B) \neq (B \circ_x A).
\]

There are several interesting classes of choice which can be characterised solely by their algebraic properties. For instance consider the class of choices \( \circ_x \) which have an inverse \( \circ_{x^{-1}} \), that is

\[
A = (A \circ_x (B \circ_{x^{-1}} A)) = ((A \circ_x B) \circ_{x^{-1}} A)
\]

\[
= (A \circ_{x^{-1}} (B \circ_x A)) = (A \circ_{x^{-1}} (B \circ_x A))
\]

\(^6\) In this paper choices are taken to be decidable choices.
Such choices are those choices which are reversible, or reproducible. These are the choices which are “deterministic” in the everyday use of the word (as opposed to the process algebra meaning). We will call choices which are not reproducible indeterminate. The choice \( o_x \) above is a reproducible choice, whereas \( o_y \) is indeterminate.

**Proposition 1** If \( o_{x-1} \) exists it will be unique and equal to \( o_x \).

Let us consider each of these two classes in turn. Reproducible choices are the traditional state based choices of programs. They can be calculated and combined using classical logic. The abstract choices of CSP are not reproducible as:

**Proposition 2** If \( o_x \) is associative and commutative it does not have an inverse.

Indeterminate choices are choices which may be made differently in two otherwise identical runs of a process. We can further divide this class of choices into those which are both associative and commutative, and those which are not. We will call the former subclass pure indeterminate choices and the latter probabilistic.

A pure indeterminate choice is one which cannot be predicted and about which nothing is known from the point of view of the process. Such choices are characterised by the processes they select between, for each unordered pair of processes there is a unique pure indeterminate choice between them. Composites of such choices are characterised by the set of possible outcomes rather than the order in which the alternatives are presented, or the manner in which the composite is constructed. The choice *select either of the processes, I don’t care which* is pure indeterminate.

An example of a probabilistic choice is \( o_y \) above.

7. A Classification of Choice

The established process models listed in §1 have differing notions of choice. These notions are typically not made explicit in the model but have to be inferred from examples provided or the denotational semantics. As yet no uniform classification of these disparate notions of choice has appeared.

It is hoped that the algebra of choice developed in this paper will enable such a uniform classification. To date the choices of CSP and finite state automata have been characterised using this algebra, and work is currently underway on CCS and Petri Nets. The results of this work will also be included in the final paper.

8. References